ETSN05 Lecture 4

Kursinfo

- Inlämning 1 på tisdag – skicka SRS, SDP, SVVS i pdf, samt var man hittar ”dokumentbibliotek” till granskaren
- Om ni inte bokat tid för granskning kontakta handledaren
- Bra att distribuera även ”utredningsuppgifter”/”expertområden” inom gruppen
- Individuell uppgift – instruktioner kommer snart

Contents

- Part 1
  - Testing
  - Risk analysis
  - Configuration management
- Part 2
  - javascript, Bootstrap

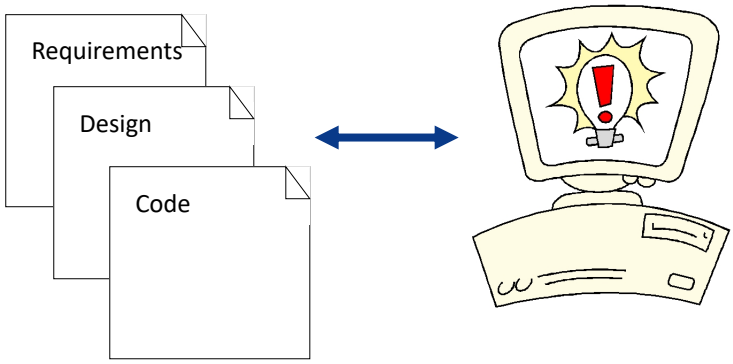
Verification vs validation

- **Verification**:
  "Are we building the product right". The software should conform to its specification.
- **Validation**:
  "Are we building the right product". The software should do what the user really requires.

4

Two approaches

- static - manual reviews
- dynamic - execution of test cases



Testing

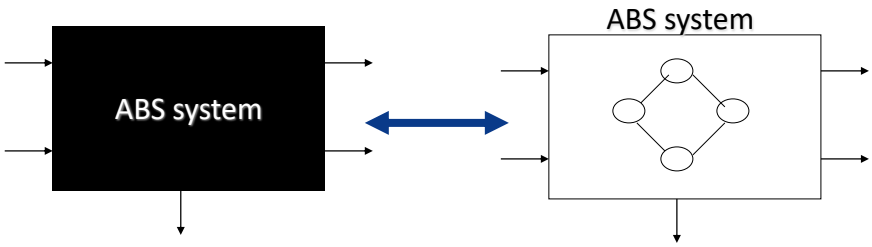
- the goal of testing is:
  - to find errors in the program
  - to build trust in the product
  - **not** to prove the absence of errors
- a large software product should be tested by:
  - the people who create it
  - an independent test group

    the test group should be independent from the implementation and base their work on the requirements specification

Testing methods

Two different points of view:

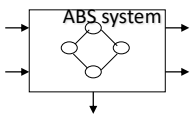
- white box - based in the internal structure
- black box - based on externally observable behaviour



Different testing techniques

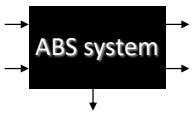
White box:

- coverage
  - statement
  - branch
  - Data flow
- ad hoc

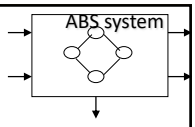


Black box:

- equivalence partitioning
- boundary values
- random testing
- error guessing
- ad hoc

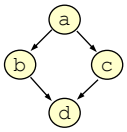


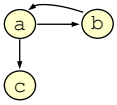
Coverage-based Testing



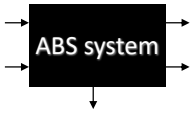
- Derive test cases from the structure of the code
  - Build the flow graph of the code
  - Cover the graph with tests as densely as possible
- Flow graphs:

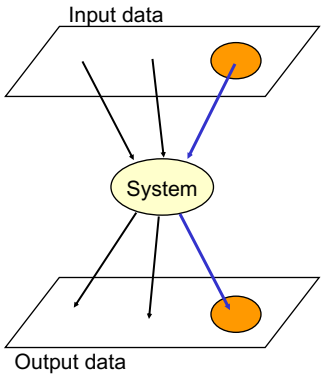
```
if a then
   b
else
   c;
d

while a do
   b;
c
```





Equivalence Partitioning / Boundary value analysis





Input- and output data can be grouped into classes where all members in the class behave in a comparable way.

➡ Define the classes
➡ Choose representatives
- Typical element
- Borderline cases

x ∈ [25 .. 100]

- class 1: x < 25
- class 2: x >= 25 and x <= 100
- class 3: x > 100

Different levels of testing

- unit test
- function test
- integration test
- system test
- acceptance test
- installation test

which done in your project?

Regression testing

- repetition of previous test cases on occasion of:
  - error correction
  - changes
  - additions to an existing system
- regression testing is quite simple:
  - you re-use already existing test cases
  - you have previous (erroneous/expected) results
- but it can be time consuming

How much regression testing should be carried out?

Testing - unit test

- **Good to automate**
- **Up to you how much you automate**
- **Early observation: The more unit tests that are done, the fewer faults will probably be found in system test**

Test process – system level

1. Identify all test cases - test specification (SVVS)
2. Describe each test case (both function test and system test) step by step in natural language - test instruction (SVVI)
3. Test and make notes of errors - and report
4. Regression test and continue testing with new test cases
5. When the software is free from errors in function- and system test, it can be approved
6. When you are ready, contact the customer for acceptance test

Things to consider

Test specification:
- short and substantial wording
- unique numbering
- cover all (functional) requirements
- reference the requirements

Test instructions:
- system state at beginning
- system state at end
- expected result
- ”self contained”

Riskhantering

- **Olika typer**
  - **Projektrisker**
  - **Produktrisker**
  - **Affärsrisker**
- **Olika allvarliga**
- **Olika risker**

Exempel på risker

- Personer slutar
- Organisationen förändras och får nya prioriteringar
- Svårt att få tillgång till hårdvara
- Utvecklingsverktyg inte bra nog
- Inköpta komponenter inte bra nog
- Konkurrent hinner före

Riskhanteringsprocess

1. Identifiera risker
2. Analysera risker
3. Planera baserat på risker
4. Följ upp

Riskidentifiering

- Gruppmöte eller av en person
- Checklista:
  - Teknikrisker
  - Risker med avseende på projektdeltagare
  - Organisationsbaserade risker
  - Verktygsrisker
  - Kravrelaterade risker
  - Skattningsbaserade risker

Exempel på risker

- Teknik
  - R1: Kommunikationslänken är för långsam
  - R2: Det går inte att återanvända så som är tänkt
- Deltagare
  - R3: Nyckelpersoner blir sjuka
  - R4: Personer måste deltaga i andra projekt
- Organisation
  - R5: Organisationen får finansiella problem och vill lägga ner projektet
- Skattning
  - R6: Underskattning av dem tid som behövs
- …

Riskanalys

Risk	p	Effekt
R1: Kommunikationslänken är för långsam	Låg	Låg
R2: Det går inte att återanvända så som är tänkt	Hög	Medel
R3: Nyckelpersoner blir sjuka	Medel	Hög
R4: Personer måste deltaga i andra projekt	Medel	Låg
R5: Organisationen får finansiella problem och vill lägga ner projektet	Medel	Medel
R6: Underskattning av dem tid som behövs	Låg	Låg

Planering

- Tre alternativ
  - Undvik problemet
  - Minimera effekten av problemet
  - Förbered en alternativ plan
- Exempel (de tre priorireade riskerna)
  - Undvik R2 (inte återanvända): Köp andra komponenter
  - Minimera effekten av R3 (sjuka): kunskaps-spridning
  - Alternativ plan för R5 (finansiella problem): förbered presentation

Uppföljning av risker

- **Kontrollera t ex med avseende på**
  - **Teknik: sena leveranser, många rapporterade problem,…**
  - **Deltagare: ”dålig stämning”, många vakanser,…**
  - **Organisation: för lite beslut från övre ledningen,…**
  - **Verktyg: låg användning,…**
  - **Krav: kundklagomål, många ändringsrapporter…**
  - **Skattning: inte klar i tid, hittade fel rättas inte,…**

Configuration Management (CM)

CM is a controlled way to manage development and change to systems during the whole life-cycle

Why do we need CM?

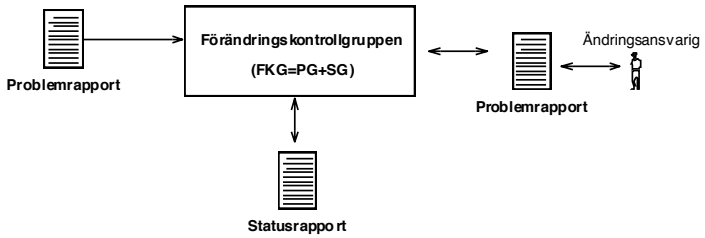
- to enable parallel work
- to know at any time what parts depend on other parts and which is the valid version
- to have a strategy for how to ”integrate” changes and corrections

Activities in Configuration Management

- identification of configuration items (CI)
- naming of versions and structuring of CIs
- change management
- status reporting
- system building - putting the parts together

Change Control Board (CCB)

- decide about changes
- the status report provides the history for each CI
- the problem report is like a baton that is passed between the different parties involved in carrying out a change and where all decisions and actions are documented

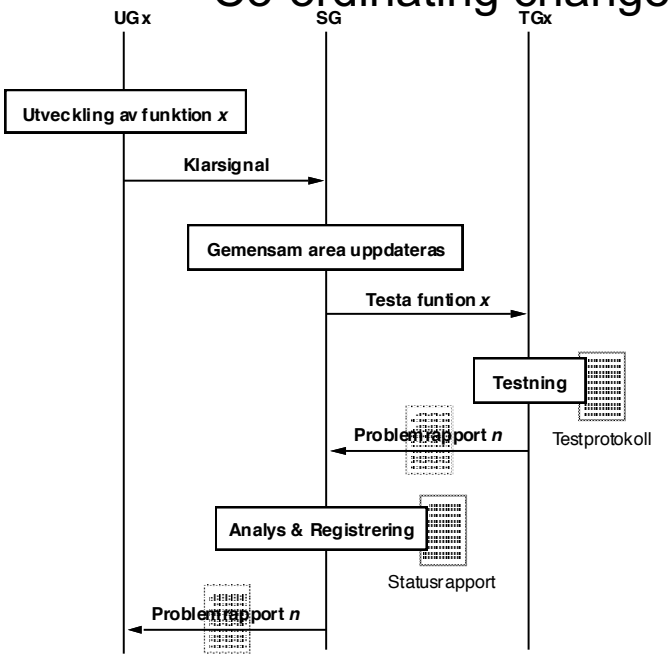


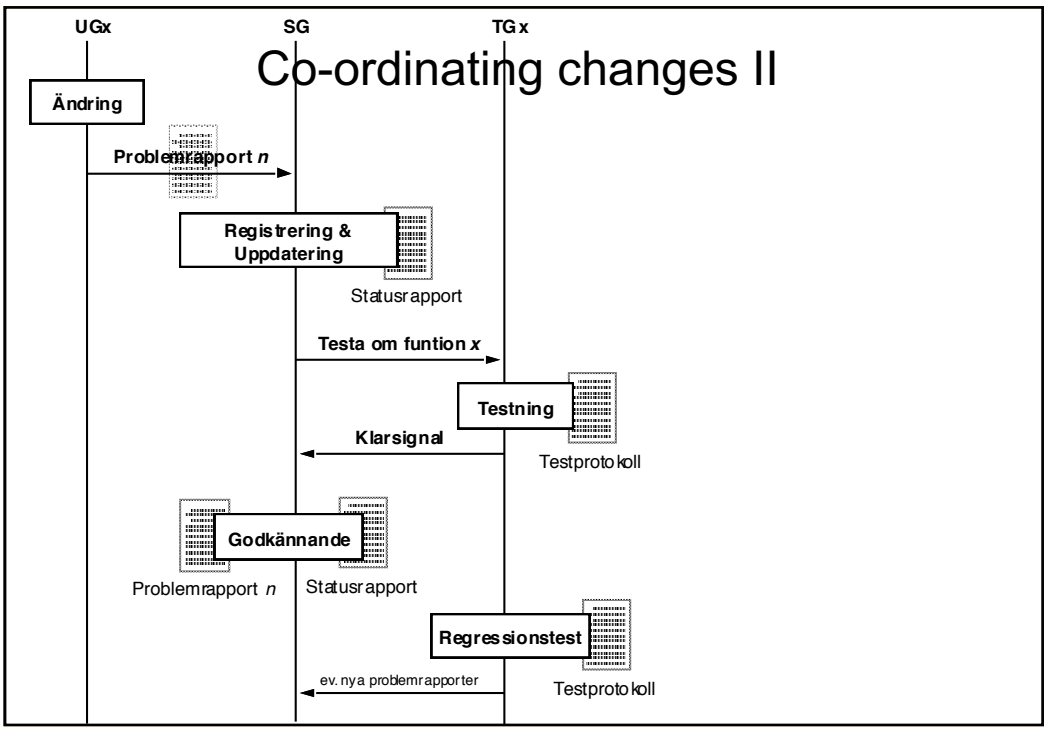
Problemrapporten skickas vidare under ändringshanteringen.

Change Management (CM;-)

- allows parallel work without the risk of destroying other people's work
- formal change management is only required for CIs that are baselined
- the baseline of the SDDD is created after (successful) informal review

Co-ordinating changes I




UGx
SG
TGx
Co-ordinating changes II
Ändring
Problemrapport n
Registrering & Uppdatering
Statusrapport
Testa om funtion x
Testning
Klarsignal
Testprotokoll
Godkännande
Problemrapport n
Statusrapport
Regressionstest
ev. nya problemrapporter
Testprotokoll